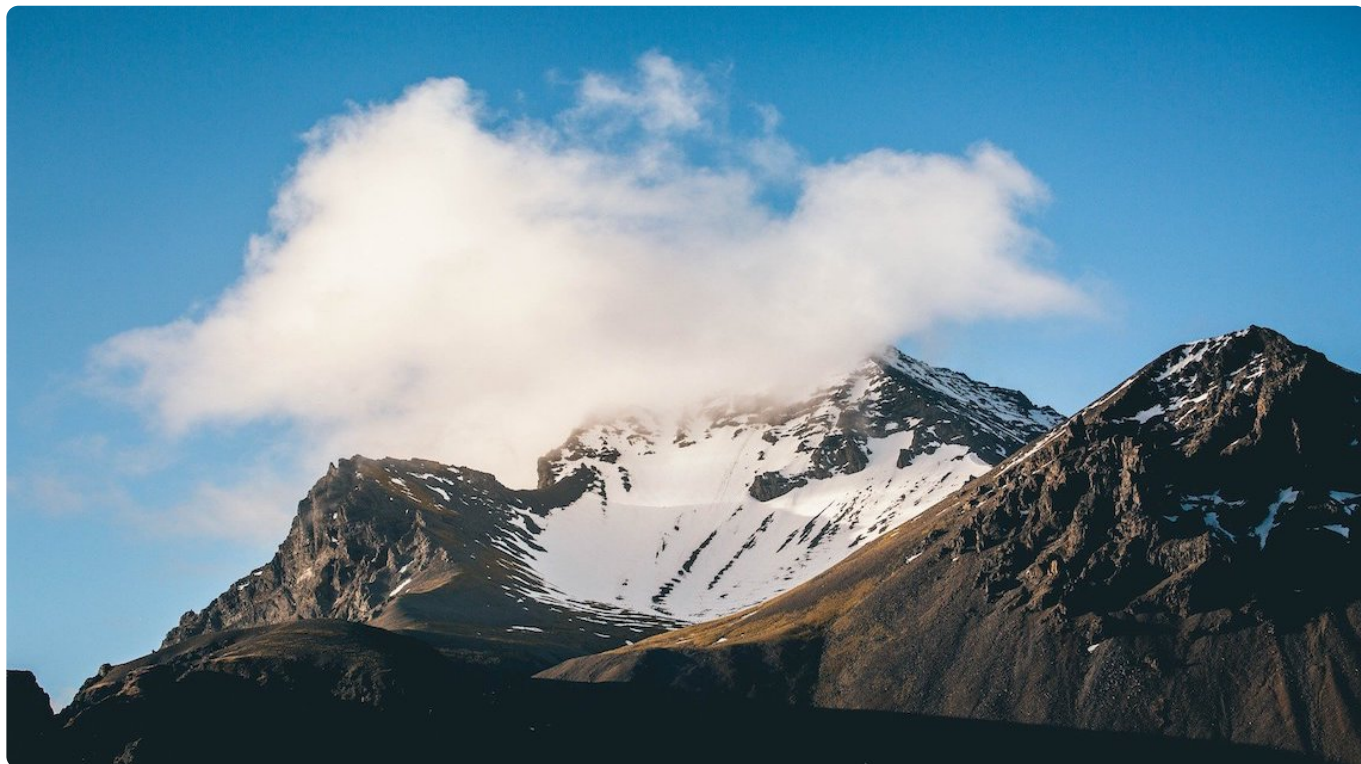


课前准备 | 搭建实验环境

2020-05-06 罗剑锋

罗剑锋的C++实战笔记

[进入课程 >](#)



讲述：Chrono

时长 08:43 大小 8.00M



你好，我是 Chrono。

在开始正式的学习之前，我们需要先做一点准备工作——在自己的电脑上搭建出课程使用的实验环境。

我会给你推荐具体的操作系统、编译器，带你一步步搭建环境，还会给你分享我的 [GitHub 链接](#)，之后课程里的所有代码，你都可以在这里找到。



操作系统

首先说一下操作系统。

目前流行的操作系统有三种：Windows、macOS 和 Linux。

Windows 是应用得最广泛的一个系统了，是绝对的主流，但是，作为 C++ 开发环境来说，Windows 并不能算是首选。

一个原因是，Windows 上的标准 C++ 开发工具 Visual Studio 不是免费的，尽管它提供了可自由下载的社区版，但有各种限制（你可以看下社区版和其他版本的 [差异](#)），用来做实验还行，如果要用来开发正式的软件，就不是那么合适了。

另一个更重要的原因是，Windows 现在已经不再是 C++ 的“主战场”了，现在开发 Windows 程序，更多的是用 C#、Java、TypeScript 等其他语言，在 Windows 上写 C++ 程序，很难有大的作为，会有种“英雄无用武之地”的感觉。

macOS 是苹果的专用系统，比较“小众精英”，用户比较少。必须要承认，它是一个很高效率易用的开发环境，但也有点“曲高和寡”，不是每个人都能有一台 MacBook 的。而且，macOS 虽然也是 UNIX，但它源自 FreeBSD，内部结构、使用方式与 Linux 有一些差异。

macOS 也有与 Windows 相同的问题，它的官方开发语言是 Objective-C、Swift，几乎没有纯粹使用 C++ 开发出的 Mac 应用。

所以，**我建议你使用 Linux 操作系统**，它是完全自由、免费的系统，不受任何人的控制，开发工具链非常完善，而且，目前差不多所有的商业网站的服务器（当然还有 Android）上跑的都是 Linux，而 C++ 也正好能在开发后台应用服务方面大显身手，两者可谓是“绝配”。

编译器

选好了 Linux 操作系统，接下来就要选择编译器了。

虽然 C++20 马上就要发布了，但现在 C++ 标准的实际普及情况还是相当落后的。据我多年的观察，很多企业因为各式各样的原因，还在用着老旧的操作系统和编译器，别说 C++20 了，连 C++17 都很少用到。

所以，从应用现状出发，我不会使用那么“超前”的标准和编译器，而是选择更贴合实际、更“接地气”的编译器，让你在工作中能够立即用得上，不用苦等操作系统、编译器的升级。

好了，说一下我对编译器的要求：**至少要支持 C++11，支持到 C++14 最好，对再往上的 17/20 则不强求，在今后的课程中，也会尽量不涉及 17/20 里的特性。**

把这几点要求落实到 Linux 上的默认编译器 GCC，就是 4.6 或者是 4.8 版本，越高越好。至于具体版本 GCC 对 C++ 的支持程度，可以在它的 [官网](#)上找到。

GCC 通常是和 Linux 系统绑定在一起的，选编译器也就相当于选择 Linux 的版本。

Linux 也是版本众多，最常见的是 **RedHat 系的 CentOS** 和 **Debian 系的 Ubuntu**。

这两个被很多企业广泛采用，但 CentOS 通常比较“稳定”，更新较慢，像 CentOS 6 一直用的是 GCC 4.4，CentOS 7 也才是 4.8，对 C++ 标准的支持很不完善，极大地限制了 C++ 能力的发挥。

所以，**我建议使用的 Linux 操作系统是 Ubuntu，最低版本是 16.04**。这个系统里的 GCC 版本是 5.4，完美支持 C++11/14。当然，你要是愿意安装更新的 18.04、20.04 也没有问题，它们里面的 GCC 版本更高，支持 C++17，只不过有点“功能过剩”。

搭建环境

确定了操作系统和编译器以后，该去哪里找一个这样的开发环境呢？

你的公司里或许就有现成的 Linux 服务器，可以直接登录上去用，但公司服务器的环境不一定满足刚才说的那几点要求，而且还得小心，别把系统搞乱了，所以，还是弄一个自己的实验环境最保险。

好在如今的虚拟技术非常成熟，只要安装一个虚拟机软件，再去 Ubuntu 官网下载一个光盘镜像，然后按部就班地点点鼠标就行了。


这里我选择的是免费的 VirtualBox，版本任意，Ubuntu 则是 [64 位的桌面版](#)。至于它们俩具体的安装过程，网上有很多资料，而且我相信，即使没有资料，也难不倒你，所以我就

不再多说了。

弄好 VirtualBox 和 Ubuntu 环境之后，还有一个小小的“收尾”步骤。

Linux 系统里通常默认只有 GCC，没有 G++，所以还要再执行一个 apt-get 命令：

```
1 sudo apt-get install g++
```

 复制代码

安装完成后，在命令行里敲一下“g++”命令，看一下它的版本号是否满足我们的要求：

```
1 g++ --version
2
3 g++ (Ubuntu 5.4.0-6ubuntu1~16.04.9) 5.4.0 20160609
4 Copyright (C) 2015 Free Software Foundation, Inc.
```

 复制代码

GitHub 项目

和之前的 [🔗 《透视 HTTP 协议》](#) 一样，我也在 GitHub 上为这个课程建立了一个项目：

[🔗 cpp_study](#)，里面有很多示例代码和有用的资料链接，你可以在 Linux 环境里用“git clone”下载。

不过，说起来不怕你笑话，我写了二十来年的 C++ 程序，但一直不怎么会写 Makefile。都说写 Makefile 是 C++ 程序员的“基本功”，但我在这方面的确是“缺失”了，有点惭愧啊。

最早，我用的是 Windows+VC 自带的工程文件；后来转到 Linux 开发，用的是 Boost 的构建工具 b2（Boost Build v2），写的是 jamfile；再后来又定制开发 Nginx，用的是 Nginx 自成体系的 Shell config。

你看，这么多年来我就基本没正经写过 Makefile，而且也没怎么用过其他的构建工具，比如 cmake、scons。

所以呢，在这个 GitHub 项目里，我也就没有办法提供专门的 Makefile，只能麻烦你在命令行上手动敲 GCC 的编译命令了。好在示例代码都很短很小，没有复杂的依赖关系，简简单单就能搞定，比如：

```
1 g++ xxx.cpp -std=c++14 -o a.out
```

 复制代码

这里需要注意的是参数 “**-std=c++14**”，它告诉编译器，在处理 C++ 代码的时候使用 C++14 标准，而不是 11/17/20。

手敲命令还是挺麻烦的，所以，我在源码文件里还用注释的形式给出了编译命令，你可以直接拷贝粘贴使用，希望能够给你带来一点点的方便。


```
1 // Copyright (c) 2020 by Chrono
2 //
3 // g++ test.cpp -std=c++11 -o a.out;./a.out
4 // g++ test.cpp -std=c++14 -o a.out;./a.out
5 // g++ test.cpp -std=c++14 -I../common -o a.out;./a.out
```

 复制代码

在目录 “section0” 里，有一个最基本的示例程序，如果你能够正确地编译并运行，就说明实验环境搭建成功了。

在我的虚拟机里，这个程序的输出是（使用 -std=c++14）：

```
1 c++ version = 201402
2 gcc version = 5.4.0 20160609
3 gcc major = 5
4 gcc minor = 4
5 gcc patch = 0
6 libstdc++ = 20160609
```

 复制代码

显示使用的是 C++14 标准，GCC 版本是 5.4.0，标准库版本是 20160609。

当然，如果你是写 Makefile 的高手，欢迎你给这个项目提 Pull Request，让其他同学都能用 make 来轻松地编译代码。

小结

作为正式开课前的“热身工作”，今天我介绍了课程使用的实验环境，简单小结一下：

1. 我们选择的操作系统是 Linux，具体是 Ubuntu 16.04，也可以是更新的版本。
2. 我们选择的编译器是 GCC，最低要求是 4.6 或者 4.8，推荐使用 5.4 或者更高的版本。
3. 使用虚拟机软件搭建环境最方便，完全“自主可控”，推荐使用免费的 VirtualBox。
4. GitHub 上有配套的示例代码和参考资料，可以下载后编译验证环境是否搭建成功。

那么，行动起来吧，下节课让我们在 Linux 上愉快地一起“玩耍”。

课外小贴士

1. 虽然在 Windows 也可以安装 MinGW+GCC，但步骤比较麻烦，费时费力。而且，既然都模拟出了 Linux 环境，为什么不一步到位，直接用 Linux 呢？
2. macOS 早期叫“Mac OS X”“OS X”（X 读作 ten），后来为了与 iOS 保持一致，才改成了“macOS”，它的内核名字叫 Darwin。
3. GCC 虽然也可以编译 C++ 代码，但它不能链接 C++ 编译单元，所以，还是统一用 G++ 比较方便。
4. 除了 GCC，另一个可选的编译器是 LLVM/Clang，但它不是 Linux 的默认编译器，所以就不过多介绍了。



上一篇 开篇词 | 把C++从“神坛”上拉下来，这次咱这么学

下一篇 01 | 重新认识C++：生命周期和编程范式

精选留言 (31)

写留言



chrono 置顶

2020-05-07

有同学已经提了一个pull request，我已经在GitHub上合并了。
不过还是有一些不足，比如用不支持C++14的gcc4.8就会make失败，大家可以再参考完善。

3

1



中年男子

2020-05-06

目前看评论，感觉好多C++萌新来这，哈哈
展开

作者回复: 这个确实有点出乎意料，可能后面会补充点更入门的文字了。

3

3



张家聚

2020-05-06

老师好，我也一直苦于不会写 Makefile，尝试几次学习写 Makefile，但查到的资料都是讲 Makefile 语法，枯燥又记不住，所以几次都放弃了。我想能不能有个稍微复杂一点的工程的 Makefile，然后附上注释，这样当我们自己构建工程的时候，就能根据这些 Makefile 文件自己改出一个适合自己工程的 Makefile，这样边用边学，可能会事半功倍。老师能不能帮我们想想办法，或者让极客时间出一个 Makefile 课程也行，我会买了学的。谢谢了。
展开

作者回复: Makefile太古老，属于比较老旧的工具链，所以才会有那么多新的构建工具，比如cmake、scons，建议简单了解Makefile就好，新的构建工具更方便好用。

3

2



reverse

2020-05-09

也可以使用vmware fusion，哈哈，我是macos系统，用这个比较有感觉，另外我装的ubuntu 16.04居然没有gcc，[狗头保命]，另外老师最后可以补充cmake的技巧吗，毕竟这个确实很常用，但是自己用的不好

作者回复: 1.我家里就是mac，以前用过VMware，但后来就转成virtualbox了，还是喜欢开源的产品。

2.Ubuntu应该都有gcc吧，也许你安装的是最小版，好在用apt可以再安装，Ubuntu的易用性还是很好的。

3.cmake我就更不会了，我最喜欢的还是Boost的构建工具b2。



1



吃草~

2020-05-07

在地铁上过一遍课程，一会儿到公司先把环境给搭好~本人使用的 macOS，从安装 VirtualBox 开始~😁😁

作者回复: 我也正在地铁上呢。



1



风

2020-05-06

不能装虚拟机，可以用cygwin吗，cygwin和mingw，哪个好

作者回复: 这两个我都没实际用过，一直是直接用虚拟机的，好像是mingw比较好，不过在Windows上模拟Linux还是有点麻烦的，毕竟不是纯粹的Linux环境。

看看吧，如果用Windows的同学比较多，我就补充一个Windows上用vs的，不过只能是现学现卖了，毕竟很久没有在Windows上写程序了。



6



ghost

2020-05-06

WSL 也是一个好的选择吗？

展开 ▾

作者回复: 应该也可以, 我没用过, 只要支持C++11/14就行, 不过后面的进阶技能要安装一些第三方库, 可能还是Linux的apt、yum比较方便。

◀ ▶

💬 1

👍 1



Why not.

2020-05-06

没有装虚拟机 电脑上装的双系统 安装的Ubuntu16.04应该也可以吧

作者回复: 当然可以, 双系统比虚拟机更好。

◀ ▶

💬

👍 1



Solomon

2020-05-06

老师, 虚拟机安装Ubuntu server版就可以了吗?

展开 ▾

作者回复: server版也可以, 不过没有图形界面, 纯命令行, 我建议学习来说用桌面版最方便。

◀ ▶

💬

👍 1



Geek_860b13

2020-05-11

完成了第一个作业, 我用4.8的bcc

展开 ▾

作者回复: 挺好的。

不过4.8还是略微落后了一些, 不支持C++14, 到后面会讲一些只有在C++14里才有的更方便的新特性, 可以有空再升级到GCC5以上。

◀ ▶

💬

👍

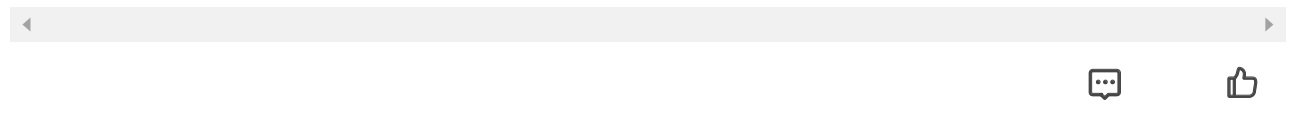


官

2020-05-11

win10好像自带了一个ubuntu的子系统，感觉可以用这个做做看看

作者回复: 嗯，好像有同学说过，这个叫WSL，可以试试，只要是Linux就没问题。



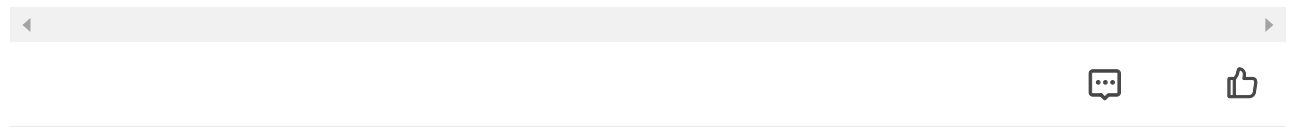
学写代码的猪

2020-05-09

deepin 的系统可以？毕竟界面比较友好。

展开 ▾

作者回复: 完全可以，只要gcc版本够高就行。

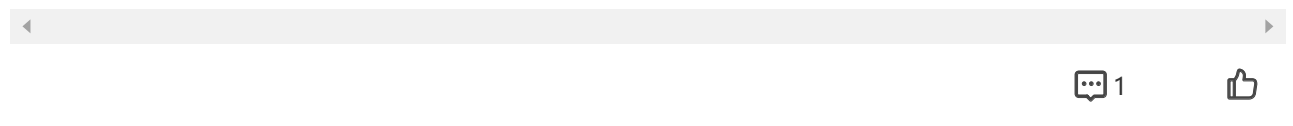


一步

2020-05-09

为什么 Mac 系统 g++ --version 和 gcc --version 打印出来的结果是一样的啊？

作者回复: 这两个是一家，当然是一个版本号了。



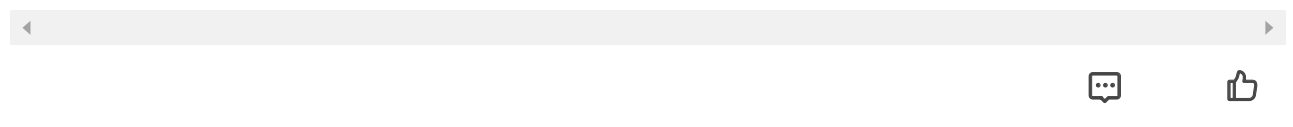
Geek_a19847

2020-05-09

Ubuntu server还是desktop不作要求吧

展开 ▾

作者回复: 是的，哪个都可以，桌面版有图形界面会更方便一些，看自己的喜好了。

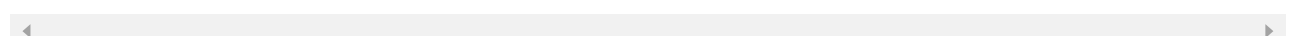


一步

2020-05-09

哈哈，为什么不用 docker 呢？想要什么环境 一个 image 搞定

作者回复: 还是有很多同学不熟悉docker吧，不过也欢迎你做出一个镜像分享给大家。





随风の

2020-05-09

其实我想说工作的时候我用的cmake很轻松=_= 简化版的makefile 而且支持跨平台 非常容易上手.

作者回复: 嗯, 我一直想学, 可惜工作中没机会, 现在用的也少了。

其实构建工具很多, 但都各有特色, 要是标准委员会定个标准的就好了, 就像Linux发明git。



乐生 ...

2020-05-08

哎。主要公司都是windows开发, 只能选择linux了, 一法通万法皆通吧。

作者回复: Linux环境比较自由, apt包管理特别方便, 对C++开发很友好。

Windows上虽然也有vcpkg这样的工具, 但还不如apt成熟。

你说的很对, 语言是和环境无关的, 只要有个好的学习环境就好。



石皮皮

2020-05-08

记得上大学的时候好像学过c++, 那时候好像用的是visual studio .竟然把这事儿给忘了

作者回复: 在Windows上vs应该是最佳选择了, 不过现在在Windows上用C++开发是少之又少, 用来学习还行。

我觉得目前C++的主力战场还是Linux。

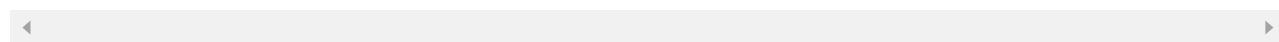


崔明

2020-05-08

安装虚拟机有点影响正常工作电脑，树莓派上安装ubuntu试过吗

作者回复: 我没试过树莓派，但只要是Linux操作系统就行，下面的硬件无所谓的。



文若

2020-05-07

直接用系统带的QT感觉也挺方便的.

展开 ▾

作者回复: 只要有自己趁手的环境就好，课程里只是抛砖引玉。

